Technical Report
1012

Data Sampling for a Burst Mode Communications System

A.G. Rocco

1 September 1994

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

Prepared for the Air Force under Contract F19628-90-C-0002.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

DATA SAMPLING FOR A BURST MODE COMMUNICATIONS SYSTEM

V.G. ROCCO
Group 47

X

By

Dist	A[illegible] or S[illegible]	
A-1		

TECHNICAL REPORT 1012

1 SEPTEMBER 1994

LEXINGTON MASSACHUSETTS

ABSTRACT

A data sampling circuit and method is provided for a burst mode communication system. The circuit is an entirely digital circuit for reliably sampling [illegible]coming stream of data to automatically adjust to variations in data stream clock rates and phase va[illegible]. the incoming data. The circuit includes a delay line with a plurality of serially coupled taps, each tap [illegible]ving a variable delay. One aspect of the invention includes increasing the delay time until the delay line captures at least one, but preferably two, full data cells of the incoming data stream (i.e., signal levels over at least one full clock period, defined by two transitions of the data stream), thereby aligning the receiving circuit with the frequency of the data stream clock. A second aspect of the invention includes outputting data from a tap that is selected to be midway between two regions of transitions of the incoming data stream, thereby aligning the receiving circuit with the phase of the data stream clock. The invention can, in alternative embodiments, track changes in the input data stream's phase/frequency. This involves updating the amount of delay in each tap of the delay line as well as picking the output of the appropriate tap to be used as the sampled data stream in response to changes in the input signal as well as changes in propagation delays of the circuits used to implement the delay line, resulting from temperature and voltage variations.

PREFACE

This document describes an invention and its various embodiments that samples an incoming serial data stream. Throughout this document the invention is referred to as the Data Sampling Circuit (DSC).

Several items in the text are followed immediately by a numeral. These numerals refer to numbers assigned in the corresponding figures cited in the text. This practice is common in patent applications, which this document was derived from.

ACKNOWLEDGMENTS

I would like to thank: my management for their continued support of my work to develop the material presented in this report; Paul Linton for writing an early version of what later became the patent application that this report is based on; and Steve Henry and John Lee, the attorneys I worked with in preparing the patent application.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

This invention was originally conceived to implement the bit-serial communication links interconnecting the individual processing nodes within the Enhanced Data Stream Array Processor (EDSAP). The EDSAP was intended to be a scalable multiprocessor system composed of multiple processing nodes, which were not necessarily identical. The EDSAP has not yet been built and it is unlikely that it will be. For an overview of the EDSAP, see Rocco, Linton, and Noonan[1].

In a computer system having multiple processors connected by a network of point-to-point bit-serial links, the point-to-point links may operate in burst mode—e.g., to save power. In this case, idle symbols are not transmitted as filler when there is no useful data to be transmitted. Accordingly, at the beginning of each newly received burst, a receiver must establish a correct phase relationship between the data stream and the receiver clock, i.e., the receiver must sample the incoming data stream at the frequency of the incoming data and within a defined range of phase to avoid an unacceptable error rate. The general approach to achieving proper phasing for sampling in the prior task is to create an internal reference clock within the receiver, synchronized to the frequency and phase of the incoming data. The process of synchronizing an internal reference clock with an incoming data stream is known as "clock recovery."

A conventional clock recovery method involves the use of a phase-locked loop (PLL). The PLL is essentially a phase detector and an oscillator in a feedback loop configuration. The phase detector compares the input signal (data) to the output of the oscillator and generates a control signal that varies with the phase difference. The phase of the oscillator's output is controlled by the output of the phase detector. The phase comparison and output phase adjustment process drives the oscillator's output to match (lock) to the phase of the input signal (data).

Until recently, PLL circuits have been implemented with analog technology. Implementing an analog PLL on an otherwise digital circuit presents several disadvantages. Early digital clock recovery circuits (sometimes called digital phase-locked loops) operate at frequencies much higher than the "recovered" clock—in fact, typically one to two orders of magnitude higher. These circuits severely limit the communication performance available from any given semiconductor technology. In the 1980s, digital clock recovery circuits operating at the same frequency as the recovered clock began to appear.

Previous PLLs (both digital and analog) that the author is familiar with in one form or another integrate phase differences between the current clock and the ideal clock, then provide a correction to move the phase of the actual clock closer to the ideal clock. Instead of responding to the average phase difference, embodiments of the subject invention—hereafter referred to as Data Sampling Circuit (DSC)—respond to peak variations in the phase of the incoming data. Because of this, the DSC can recover data much better than current methods from data streams where the distribution of the phase deviations is not "nice," e.g., a Gaussian or uniform distribution. U.S. patents 4 789 996, 4 819 251, and 5 040 193 [3,4,5] give examples of recent digital PLLs along with some of the background cited here.

The DSC provides an all-digital data sampling circuit and related sampling method for a burst mode data communication system, using a digital delay line to capture a portion of the data stream. The delay line includes a fixed number of taps, each tap providing a variable delay time. One aspect of the DSC involves progressively increasing the total delay time of all delay elements until the delay line captures at least one, but preferably two, full data cells of the incoming data stream (i.e., signal levels over at least two full clock periods, defined by transitions of the data stream), thereby adjusting the length of the delays used in the receiving circuit to coincide with the frequency of the data stream clock. In this document a data cell is defined as a portion of the incoming signal that is bounded by regions where a transition may occur. For example: with 4/5 bit encoding, four bits of data are encoded into five data cells; with Manchester encoding, one bit of data is encoded into two data cells. A second aspect of the invention includes outputting data from a tap that is selected to be midway between groups of taps in the delay line where transitions occur, thereby aligning the data sampling circuit with the phase of the data and indirectly with the phase of the data stream's clock. This process can be thought of as acquiring the input data stream. A third aspect of the DSC is the ability to track changes in the input data stream's phase/frequency.

The DSC is intended to be used, for example, in a massively parallel processor or data network, each with a globally distributed clock. In this situation, the DSC first acquires and then tracks the input signal. Updates to the tap picked as the sampling point are made just after the end of a frame of data, in such a way that no data is lost. The DSC is further intended to be used in point-to-point links where the sender sends bursts of data, each with approximately the same phase as the previous one. An embodiment of the DSC takes advantage of this by "remembering" the phase from the previous burst, using the same tap selection for the next burst. This permits the use of very short preambles (data at the beginning of a frame provided for synchronization only; it does not contain information that is actually being used).

The DSC has also been shown to work well in systems where a clock is not globally distributed. Some possible variations to the DSC will be discussed in the remainder of this report.

2. DETAILED DESCRIPTION

The DSC provides a method, and an all-digital circuit for practicing the method, to reliably sample an incoming stream of digital data. It automatically adjusts to variations of data stream clock rates and phase variations in the incoming data. Key portions of the circuit of the present invention operate at the same clock rate as the clock rate of the incoming data, so it does not impose design constraints that limit the performance of the system as a whole. The circuit initializes the link by providing frequency and phase synchronization on power-up and maintains synchronization once initial synchronization has been accomplished.

In the DSC, the data stream clock is assumed to accompany the data. Although the phase of the clock with respect to the data does not need to be controlled, it is assumed that one cycle of the data stream clock equals one data cell of the incoming data. Three ways the clock might be distributed are:

1. By globally c .ributing the data clock to all devices that are receiving data. Note that if the same device is receiving multiple data streams, it still only requires one clock (assuming that all the data streams are at the same data rate).
2. By adding a third, one-way link for the clock when using a bidirectional point-to-point link. It does not matter which station originates the clock so long as both stations are using the same clock. The DSC makes it unnecessary to control the phase of the clock so long as the phase does not change rapidly.
3. By providing oscillators to both the sender and receiver that are approximately at the same frequency and using them to clock the data.

An important concept of the DSC is to "capture" at least one full data cell of the transmitted data stream within a tapped delay line. In a described embodiment, the number of taps is fixed. Compensating for frequency differences involves adjusting the tap delays until at least one, but preferably two, data cells are contained within the delay line. This process of adjusting for the incoming frequency also adjusts for variations in the propagation delays of the delay elements used, fabrication variation, and temperature and supply voltage changes as well as variation in the data rate of the incoming data stream. Compensating for phase differences involves detecting regions in the delay line where the inter-data-cell transitions are occurring and selecting the output of a tap that is approximately midway between two regions of transitions; this allows the incoming stream to be sampled at approximately the midpoint of each of the incoming data cells.

In a local area network or within a multiprocessor system, where the same clock is globally distributed and the communication links are all point-to-point links, the phase of the data with respect to the clock will, generally, not change very much from one burst to the next. An embodiment of the DSC takes advantage of this by remembering the phase from the previous burst and using it for the next. This permits the use of very short preambles. A unique feature of the DSC is that it remembers phase information from one burst to the next.

Instead of responding to the average phase difference, embodiments of the DSC respond to peak variations in the phase of the incoming data. Because of this, the DSC recovers data well from data streams where the distribution of the phase deviations (jitter) is not "nice," e.g., a Gaussian or uniform distribution

Figure 1 shows an overall block diagram of a exemplary data sampling circuit according to the DSC. Figure 1 shows a digital delay line 25, an array of data sampling D flip-flops 27, a data selection multiplexer 23, and control logic circuit 24. In a typical embodiment, the delay line 25 has, for example, 24 programmable delays 25_0 ... 25_{N-1} (where N = 24 in this example). The received data stream enters the delay line on line 10. The length of the delay for the programmable delays is set by the control signal on control line 26. In an exemplary embodiment, tap control line 26 carries 11 bits of data to pick one of 11 possible delay settings. In the illustrated embodiment of the DSC, the same delay value is set in each of the N taps. The output of each of the programmable delays 25_{k-1} (where k = 1 ... N) is sampled by its corresponding D flip-flop 27_k (there is also a D flip-flop 27_0, which samples the input to 25_0); these D flip-flops 27_i (where i = 0 ... N) are clocked by the data clock supplied on line 11. The output of every D flip-flop 27_i is carried on corresponding lines 28_i to both the logic control circuit 24 and the multiplexer 23.

Notice that in the embodiment shown in Figure 1, the programmable delays 25_{k-1} each invert the data stream as it propagates through them; any effects due to asymmetry between rising and falling signals being propagated by the circuits used to create the programmable delays will not be additive as the data propagates through the delay line. Compensation for this inversion is provided by having every other D flip-flop 27_i provide an inverted output.

Figure 2 shows a diagram of the logic used for the programmable delays 25_{k-1} in an embodiment of the DSC. Table 1 gives typical propagation delays for each of the circuit elements used in the programmable delay shown in Figure 2. In a typical integrated circuit process in which the illustrated embodiment might be fabricated, the minimum delay might be as little as .65 of the typical and the maximum delay might be as much as 1.8 of the typical due to variations in process, temperature, and power supply voltage. Line 30 is the input to the delay and line 35 is the output. The delay is programmed by having a single one of the eleven delay enable lines 26 [i.e., DE(10:0)] true (i.e., equal to logical one). If the minimum delay is desired, DE(0) is made a one. If the maximum delay is desired, DE(10) is made a one. Table 2 gives the typical delays that are achieved by this embodiment.

Figure 3 shows a block diagram for an exemplary implementation of the control logic circuit 24. In the illustrated embodiment, the control logic circuit includes an array of XOR gates 41, an array of type JK flip-flops 43 and an array of D flip-flops 46 feeding a state machine 47. The JK flip-flops 43 are clocked by the data stream clock 11. The XOR gates 41_0 through 41_{N-1} output a one when there is a transition between consecutive data cells containing different values in the programmable delay (at the time the D flip-flops 27_0 through 27_N were clocked) whose input and output are being compared by the XOR gate. The output of this XOR gate then goes to a corresponding J input of the corresponding gate of JK flip-flops 43_0 through 43_{N-1}. These JK flip-flops are used as transition flags. If there was a transition present in the corresponding programmable delay 25_0 through 25_{N-1} (see Figure 1), the JK flip-flop (transition flag) becomes set. Once set, it will stay set until cleared by the state machine 47 (Figure 3) using the signal on line 48. For example, if there is a transition in programmable delay 25_0 at the time the D flip-flops 27_0 through 27_N are clocked, transition flag flip-flop TF0 (43_0, Figure 3) will become set. The output of the JK flip-flops 43_0 through 43_{N-1} feed an array of D flip-flops 46_0 through 46_{N-1}. These D flip-flops sample the output of the JK flip-flops at the clock rate of the state machine 47. In the illustrated embodiment, the state machine is clocked at one-tenth the rate that the data is clocked. The divider 44 divides the data stream clock on line 11 by a factor of ten, creating the clock for the state machine, signal 45.

In Figure 1, the output of each of the programmable delays 25_{k-1} is sampled by its corresponding D flip-flop 27_{k-1} to produce an output that is fed simultaneously to the control logic circuit 24 and the multiplexer 23. In the first mode of operation, the control logic circuit increases the programmable delay values via line 26 until at least one (preferably two) data cells are in the delay line and then picks which programmable delay output should be used as the sampled data stream. This mode of operation is active during power-up initialization and also during fault recovery when necessary—for example, if the transmission medium was unavailable too long for the second mode of operation to track the variation in either frequency or the change in propagation delay of the circuits within the data sampling circuit. The second mode of operation tracks changes in the phase of the incoming data and also the variation in either frequency or the change in propagation delay of the circuits within the data sampling circuit. The first mode can be thought of as acquiring the incoming signal and the second mode can be thought of as tracking the incoming signal.

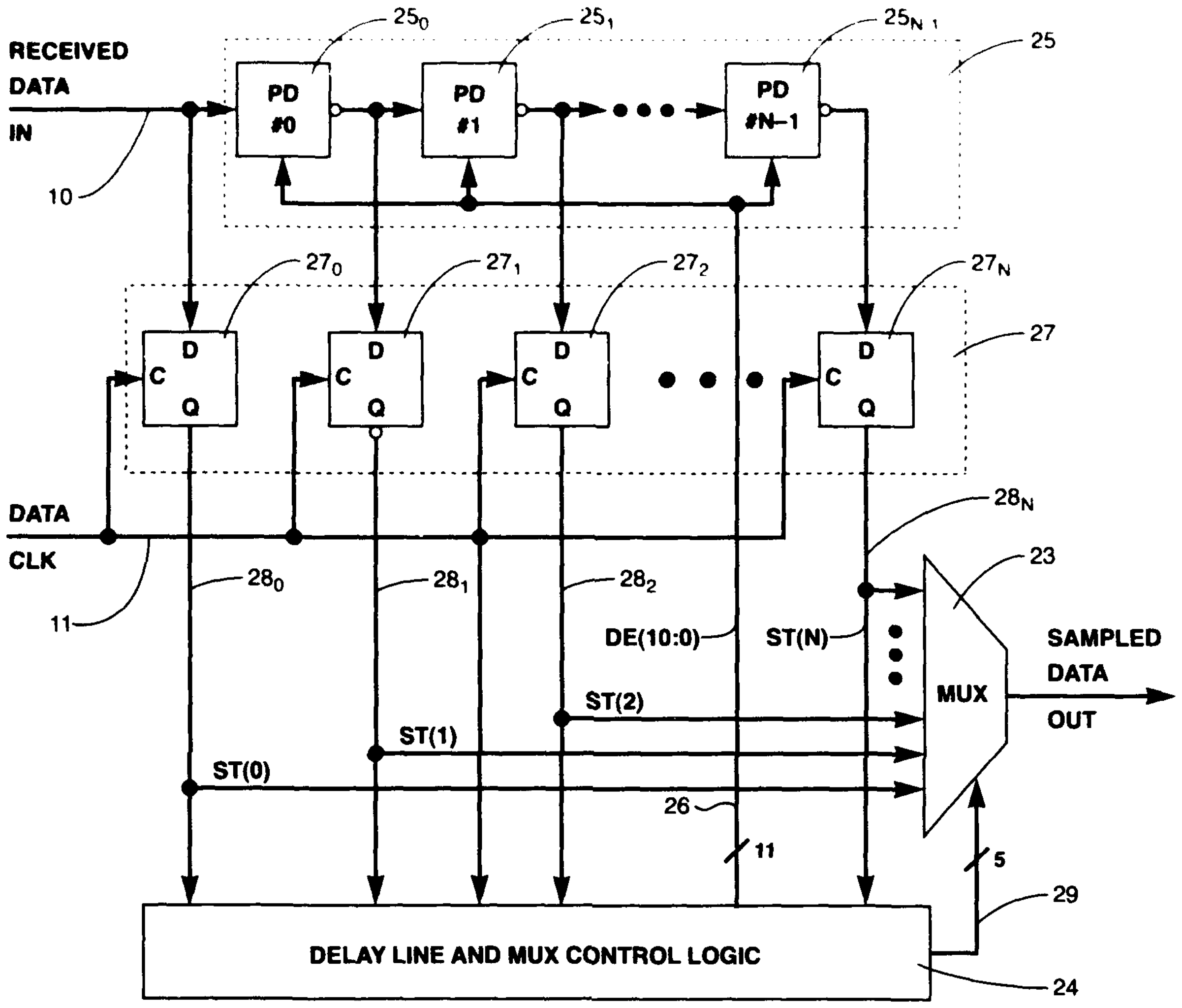


Figure 1. Data sampling circuit.

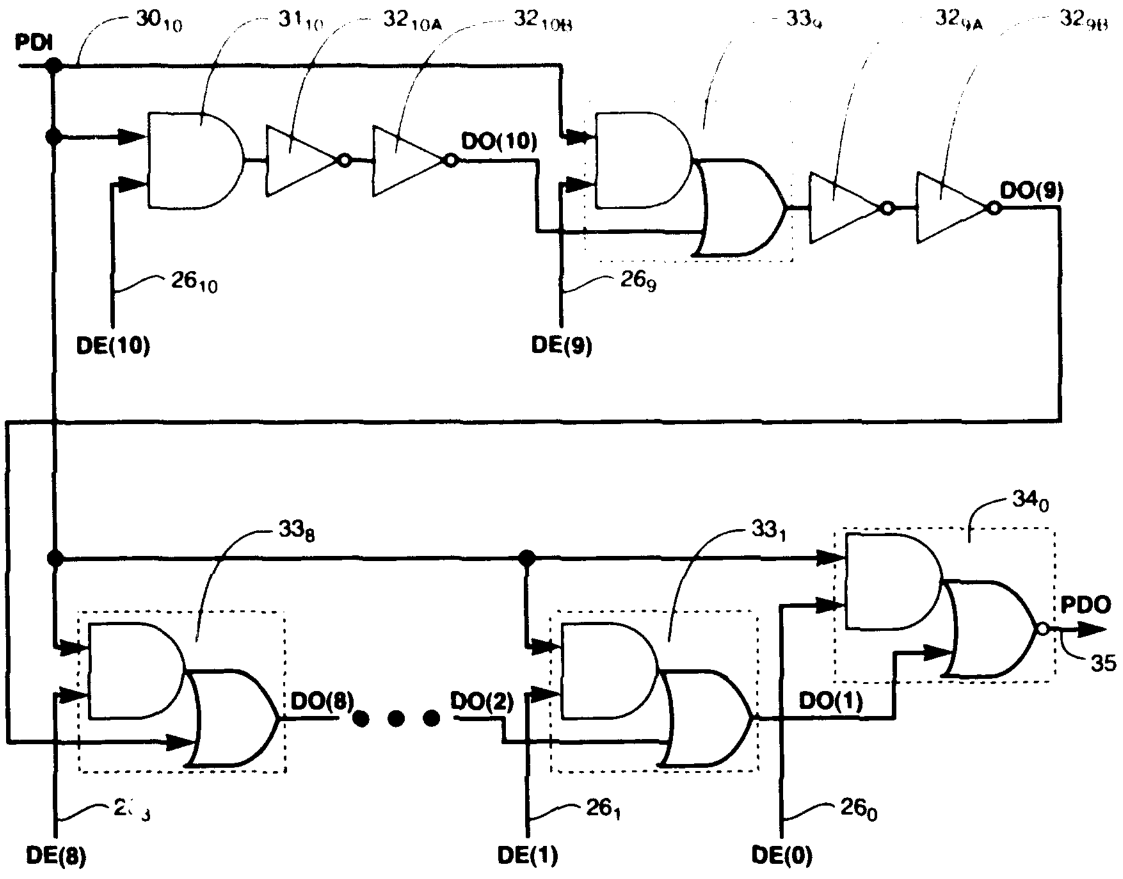


Figure 2. Programmable delay.

TABLE 1
Typical Propagation Delay for Circuit Elements Used in Programmable Delay

Circuit #	Delay (ns)	Description
31_i*	.31	2-input AND gate
32_i*	.12	inverter
33_i*	.32	gate ORing the AND of two inputs with a third input
34_0	.40†	gate ORing the AND of two inputs with a third input, with the output inverted

* "i" is simply used as a general place holder for a subscript.

† Includes .2 ns of delay to account for capacitive loading.

TABLE 2
Typical Delays for Programmable Delays Used in the Illustrated Embodiment

N	DE(10:0)	Delay in ns				Frequency (MHz)*
		Total	Change	N / (N–1)	N / (N+1)	
0	000 0000 0001	0.40	.40		0.56	250.0
1	000 0000 0010	0.72	.32	1.80	0.69	138.9
2	000 0000 0100	1.04	.32	1.44	0.76	96.2
3	000 0000 1000	1.36	.32	1.31	0.81	73.5
4	000 0001 0000	1.68	.32	1.24	0.84	59.5
5	000 0010 0000	2.00	.32	1.19	0.86	50.0
6	000 0100 0000	2.32	.32	1.16	0.88	43.1
7	000 1000 0000	2.64	.32	1.14	0.89	37.9
8	001 0000 0000	2.96	.32	1.12	0.84	33.8
9	010 0000 0000	3.52	.56	1.19	0.86	28.4
10	100 0000 0000	4.07	.57	1.16		24.6

* The data clock frequency that will cause baud times, within the delay line, to be 10 taps in length—1/(10 X delay).

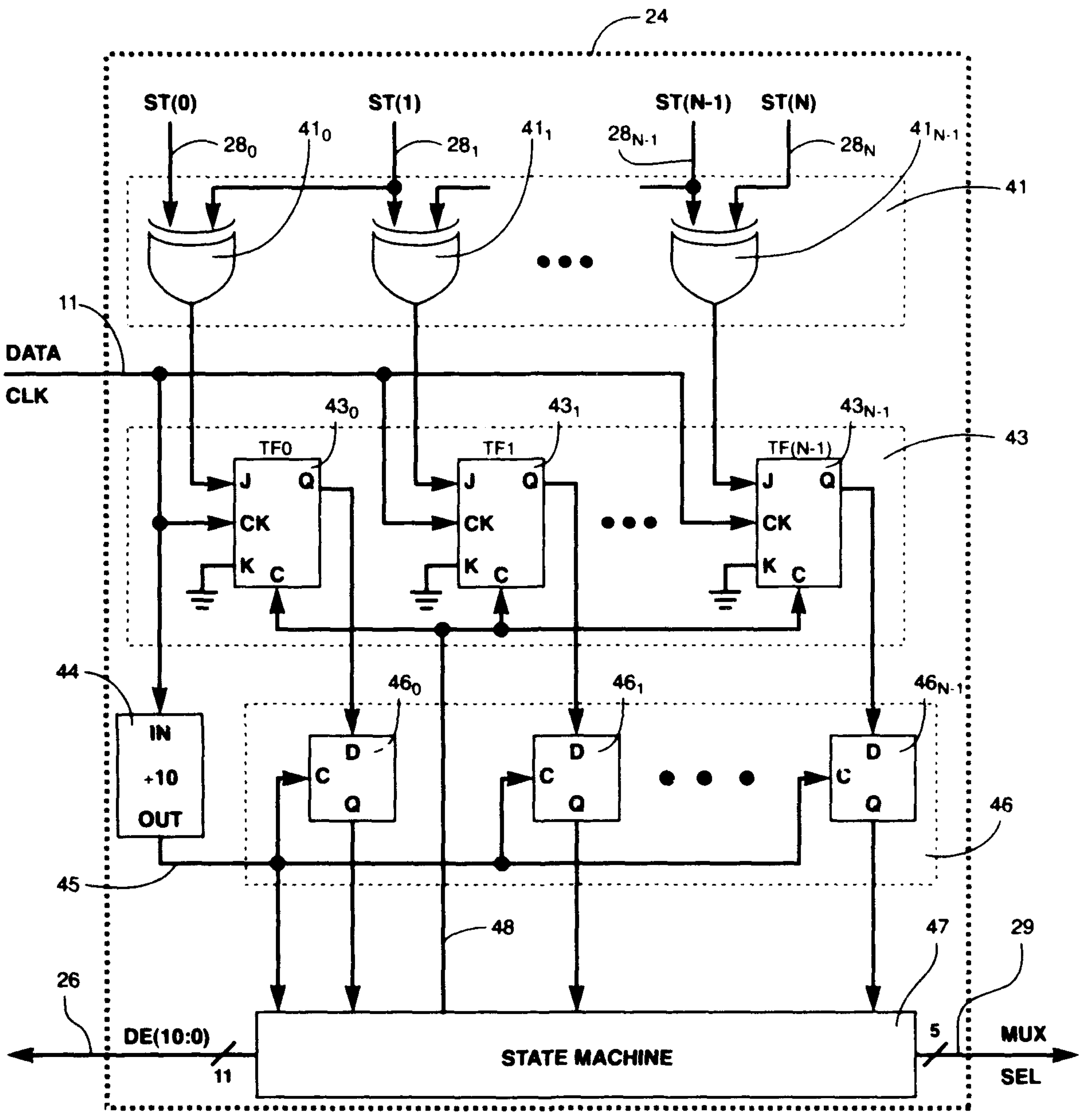


Figure 3. Delay line and multiplexer control logic.

2.1 ENCODING AND PROTOCOL USED BY THE ILLUSTRATED EMBODIMENT

Tables 3 and 4 give some of the registers associated with the receiving and transmission of data. TCERRCNT, TOK, RTAPDLY, RTAPSAM, RINCPD, RDECPD, RDCLEN, RACNT, RATF, RCERRCNT, and ROK are generally updated by the state machine 47 (Figure 3) and are not generally written to directly. The other registers are options that affect the behavior of the illustrated embodiment. Their behavior will be discussed.

The illustrated embodiment uses the same encoding scheme as FDDI (fiber distributed data interface [2]), which is 4/5 bit encoding so that a byte of eight bits of data is encoded into 10 data cells for transmission. As with FDDI, the ten data cells are further encoded into NRZI (Non-Return-to-Zero, Invert-on-ones) format.

In such an embodiment, the data being transmitted over the communication channel is formatted in frames as shown in Figure 4 and Figure 5. Referring to Figure 4, a general communication frame consists of seven fields: a preamble (PRE), a starting frame delimiter (SFD), a header (HDR), an address (ADR), a descriptor (DES), a data field (DATA) and a frame check sequence (FCS). The number of bytes in each field are: PRE, variable; SFD, 1; HDR, 6; ADR, 2 to 130; DES, 2; DATA, 0 to $\text{Frame\_Length}_{max}$ - length_of_ADR_field - 13; FCS, 4. The preamble is the first field to be transmitted, followed by the SFD and continuing from left to right, as shown in Figure 4, until the FCS is transmitted. The most significant byte within a field is transmitted first, with the most significant bit within each byte transmitted first.

Figure 5 shows the ACK frame format. The header of both the ACK and general format frames include a two-bit field, called ACKOP, which can have any of the following values: 0 = NOP (no-operation), 1 = ACK (positive acknowledgment), 2 = NAK (negative acknowledgment), 3 = UPD (the first frame of an update programmable delays frame sequence). There is also a bit in the header of both frame types that is a copy of the transmitting end's ROK bit. If the ROK bit is on, it indicates that the local end of the link is receiving properly.

The length of the preamble is variable. It should be set to at least two bytes—four symbols where each symbol is composed on five data cells. These two bytes will give a time during which the multiplexer 23 can be switched without lost of actual data; the time provided by these bytes is only used for switching the multiplexer when frames are back-to-back. The SFD symbols are used to establish byte synchronization. The number of bytes for the preamble is set by the variable TPRELEN (Table 3) so that it can be made higher for some transmission media. For example, some fiber-optic systems may require several bytes before they will properly transmit data.

TABLE 3
Transmit Status/Control Registers in the Illustrated Embodiment

Mnemonic	# Bits	Description
TXIDLE	1	Transmit idle symbols when the link is idle
TPRELEN	4	Length of preamble in bytes, default is 2
TUPCNT	8	Number of frames transmitted as part of an "update programmable delays frame sequence"
TLTO*	8	Transmit link time out—if do not receive a frame within this interval, turn off TOK
TTO*	8	Transmit time out—if do not receive an ACK or NAK within this interval, consider it a time out
TCERRCOK	8	Number of consecutive NAKs/TOs before TOK cleared
TKAINTV*	8	Interval between sending keep-alive frames
TAQINTV*	8	Interval between bursts of frames sent to facilitate reacquisition
TAQLEN*	8	Duration of burst of frames sent to facilitate reacquisition
TCERRCNT†	1	Number of consecutive transmit errors that have occurred
TOK†	1	Transmit link OK

* Time interval is equal to 163,840 cycles of the data clock 11 multiplied by the number specified.
† This register is written to by the state machine.

TABLE 4
Receive Status/Control Registers in the Illustrated Embodiment

Mnemonic	# Bits	Description
RADCNT	8	Accumulate done count—number of bytes to accumulate over to-find regions of transitions
RADCNTCD	8	Same as RADCNT except used for first accumulation after changing RTAPDLY
RWUPD	1	Wait for "update programmable delays" frame before updating the programmable delays
RCERRCOK	8	Number of consecutive RCV errors before clear ROK
RTO*	8	RCV time out—if no frames for this interval clear ROK
RTAPDLY†	4	Delay setting for each tap of delay line
RTAPSAM†	5	Tap of delay line being sampled by multiplexer 23
RINCPD†	1	Flag that indicates increment the programmable delays
RDECPD†	1	Flag that indicates decrement the programmable delays
RDCLEN†	5	Length of data cell in terms of delay line taps
RACNT†	8	Number of bytes have accumulated over
RATF†	24	Accumulate transition flags register
RCERRCNT†	8	Number of consecutive RCV errors that have occurred
ROK†	1	Receive link OK

* Time interval is equal to 163,840 cycles of the data clock 11 multiplied by the number specified.
† This register is written to by the state machine.

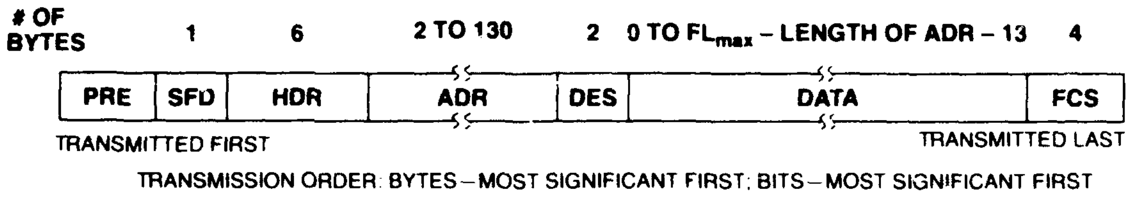

# OF BYTES
1
6
2 TO 130
2
0 TO FL_max – LENGTH OF ADR – 13
4
PRE
SFD
HDR
ADR
DES
DATA
FCS
TRANSMITTED FIRST
TRANSMITTED LAST
TRANSMISSION ORDER: BYTES—MOST SIGNIFICANT FIRST; BITS—MOST SIGNIFICANT FIRST

Figure 4. General frame format.

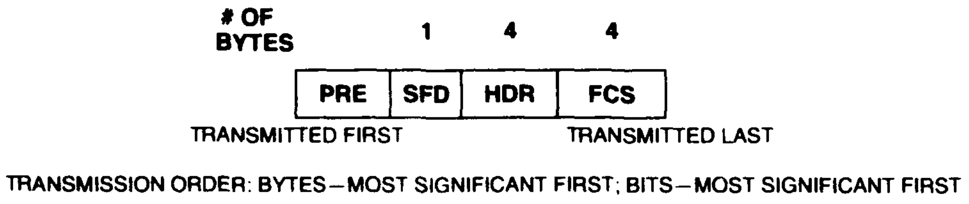

# OF BYTES
1
4
4
PRE
SFD
HDR
FCS
TRANSMITTED FIRST
TRANSMITTED LAST
TRANSMISSION ORDER: BYTES—MOST SIGNIFICANT FIRST; BITS—MOST SIGNIFICANT FIRST

Figure 5. Acknowledge frame format.

2.2 ACQUIRING THE PHASE AND FREQUENCY OF THE INCOMING SIGNAL

Figure 6 shows how an incoming waveform might be in the tapped delay line. In this example the programmable delays are set to 2 (using, for example 1.04 ns per tap). The actual waveform 64 shows the resulting reception of a large number of data cells. The transition region between cells represents the range of a substantially random distribution of real transition occurrences about an expected transition occurrence position. In a perfect transmission system, transitions, when they appear, would always appear at an expected time and would therefore occupy an expected position in the delay line, as shown in the waveform 62. However, in any practical system, the expected position and the "region of transitions" represent a range of positions for what would ideally be a single position defining the best position of a single tap. The transition flags that would become set in response to these regions of transitions are shown as the flags 61.

Figure 7 gives the process executed by the state machine during its acquisition mode, which is used during initialization and fault recovery. Steps 200 through 300 adjust for the frequency of the incoming data stream and the propagation delay of the circuit elements used to implement the data delay line 25 (Figure 1). The state machine starts with the programmable delays set to their minimum values and increments them up until at least two full data cells are in the delay line (RDCLEN less than or equal to 12). Incrementing, as opposed to a binary search, minimizes the possibility of aliasing the data (a condition that occurs when multiple transitions occur within one programmable delay). Figure 8 shows which transition flags become set for different settings of the programmable delays 25_0 through 25_{N-1} (Figure 1). This figure will be discussed in more detail.

Steps 210 through 280 (Figure 7) find out how many programmable delays of the tapped delay line 25 (Figure 1) a data cell occupies. To do this, step 220 "accumulates" data over many incoming data cells. Due to noise, intersymbol interference, and other effects, the transitions between one data cell and the next will not always happen at the same locations in the tapped delay line 25 (Figure 1). By accumulating over many incoming data cells, regions where transitions are occurring are found. This accumulation takes place in the transition flags TF0 through TF(N−1) (43_0 through 43_{N-1}, Figure 3). The number of incoming data cells that should be accumulated varies with the nature of the communication channel being used. In the illustrated embodiment, the number of byte times of the accumulation interval is programmed using the register RADCNTCD (receive accumulate done count after changing delays).

In this embodiment, idle symbols may or may not be transmitted when the link is idle, depending on the transmission medium. If idle symbols are transmitted between frames, they, as well as any frames received, are used for finding the transition regions. If idle symbols are not transmitted between frames, the first TPRELEN (Table 3) byte times worth of data are not used for accumulation of regions of transitions; ignoring the first TPRELEN bytes allows the channel and receiver to reach equilibrium after a period of receiving nothing.

Once the accumulation is finished, step 230 (Figure 7) copies the transition flags 43_0 through 43_{N-1} (Figure 3) into the internal register RATF (accumulate transition flags) of state machine 47. It is possible that a few transition flags will not be set within a transition region; steps 240 and 250 take care of this problem. Step 240 finds the largest region of zeros (region with no transitions); step 250 then deletes any regions of zeros that are too small, making them part of a region of ones (a region where transitions are occurring). In an embodiment described here, this is done in accordance with Table 5.

Steps 260, 270, and 280 (Figure 7) find the length of a data cell in terms of the number of programmable delays occupied by a data cell. This is done by measuring the distance between the centers of regions where the transition flags are zeros (not set) or regions where they are ones (set). The first full region (a region not cut off by the end of the delay line) of either zeros or ones is used in making this measurement. It is assumed that the first region is cut off, making the second region the first full one, so the distance between the second and fourth regions is measured. The resulting measurement is put into the register RDCLEN (data cell length), which is a register internal to the state machine 47 that holds the value of a parameter named RDCLEN. If there are not two regions of the type being measured, it is assumed that the programmable delays 25_0 through 25_{N-1} (Figure 1) are currently too short and RDCLEN is set to 24 so that steps 290 and 300 will increase the delay of the programmable delays.

Step 290 looks at RDCLEN (i.e., the parameter value). If RDCLEN is 12 or less, the goal of having at least two full data cells in the delay line 25 has been accomplished and the state machine goes on to step 310. Otherwise, the settings for the programmable delays are increased by one—assuming they can be—and the state machine goes back to step 300 followed by step 210. If the programmable delays are already at their maximum, the state machine goes on to step 310 regardless of the value of RDCLEN.

Step 310 (Figure 7) selects the programmable delay's output (i.e., the tap) that corresponds to the center of the largest region of zeros (i.e., region with no transitions) to be used as the sampled data stream. This is done by setting the value of the state machine's internal receive tap sample register RTAPSAM (Table 4), which in turn causes multiplexer select lines 29 (Figure 1 and Figure 3) to choose the appropriate D flip-flop output 27_i for output from the multiplexer 23.

With the completion of step 310, the process has now been completed and the state machine begins execution of the process shown in Figure 11, Figure 12, and Figure 13 to track changes in the frequency or phase of the incoming signal as well as any changes in the propagation delay of circuit elements used to make up the programmable delays 25_0 through 25_{N-1} (Figure 1). The process of tracking the incoming signal will be discussed in Section 2.4.

Figure 8 gives an example of the process of initially setting the programmable delays (Figure 7 and Table 5). In this example, it is assumed that the data stream clock rate is 100 MHz and that the propagation delay of the circuit elements used to make up the programmable delays are their typical values as given in Table 1. Table 2 gives the propagation delay of each programmable delay as a function of the setting of the programmable delays. The process of finding the proper setting for the programmable delays is as follows:

1. The state machine starts off with the programmable delays set to their shortest setting. This gives each of them a known delay of, for example, .4 ns. With a data stream clock of 100 MHz, a single data cell is 10 ns in duration. With the programmable delays set at 0 (.4 ns), a data cell will be 25 taps of the delay line 25 in length. This is too long to be measured by the tapped delay line as shown by the transition flags 71 (Figure 8). The state machine will increase by one the setting of the programmable delays.

2. With the programmable delays set to one, which corresponds to .72 ns per tap, 72 shows the centers of the regions of ones at 4.5 and 18.5; 18.5 – 4.5 = 14, indicating that a data cell is 14 taps long. Because this is greater than 12, Steps 290 and 300 (Figure 7) of the state machine will increase the setting of the programmable delays by one.
3. With the programmable delays equal to two, which corresponds to 1.04 ns per tap, a data cell will be 13 – 3.5 = 9.5 taps long, as shown by 73. This is less than or equal to 12, so the state machine will leave the programmable delays set to two.
4. The output 28_8 is at the center of the largest region of zeros, so the state machine will use the multiplexer 23 to select it as the one to sample for the data stream output.
5. The state machine has now completed acquisition, so it is switched to the tracking mode.

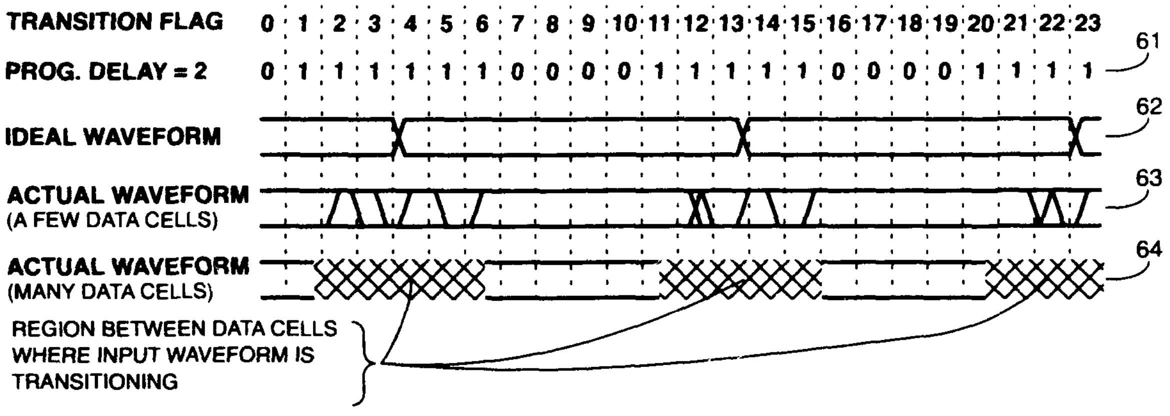


Figure 6. Relationship between transition flags and received waveform.

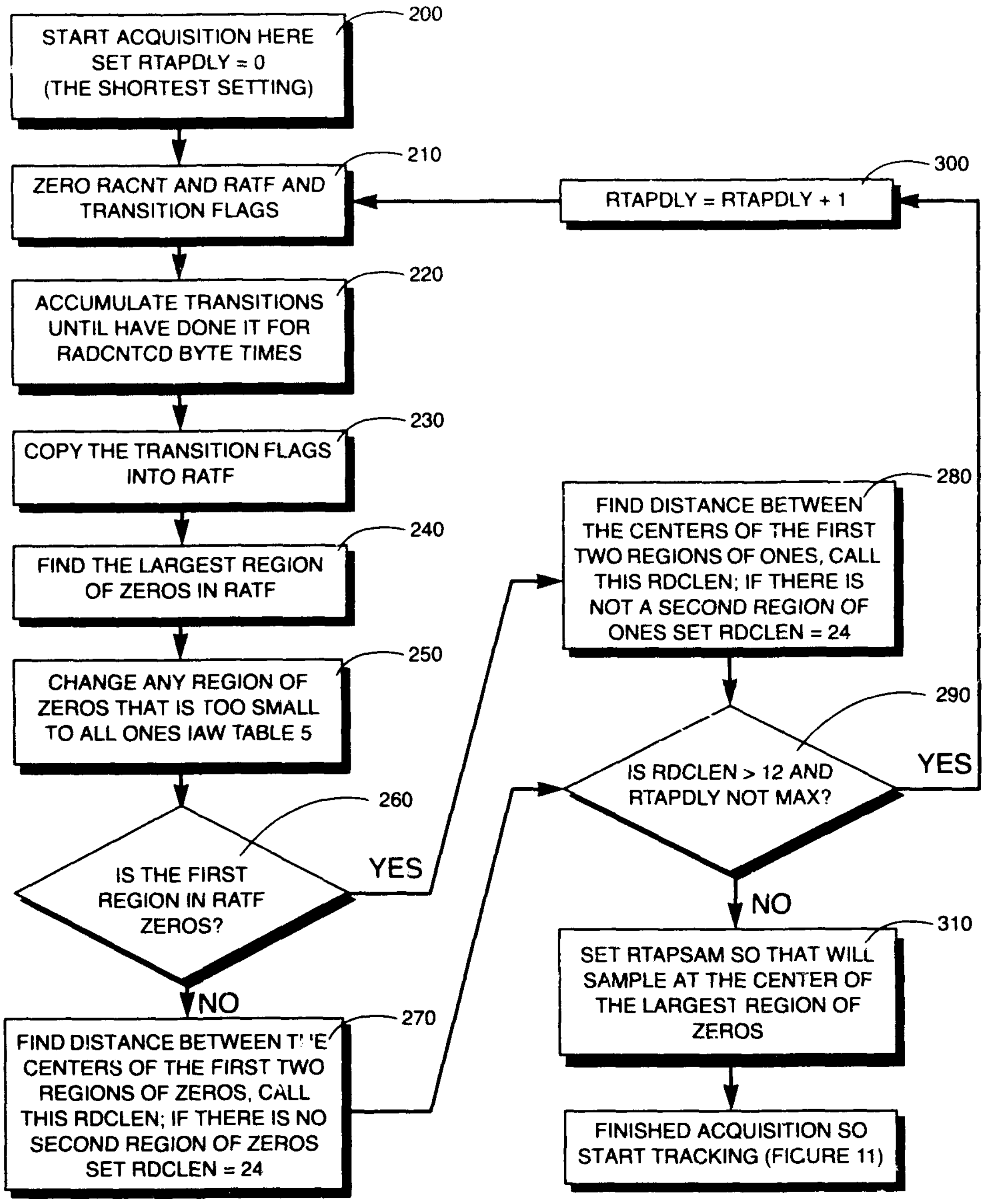

200
START ACQUISITION HERE SET RTAPDLY = 0 (THE SHORTEST SETTING)
210
ZERO RACNT AND RATF AND TRANSITION FLAGS
220
ACCUMULATE TRANSITIONS UNTIL HAVE DONE IT FOR RADCNTCD BYTE TIMES
230
COPY THE TRANSITION FLAGS INTO RATF
240
FIND THE LARGEST REGION OF ZEROS IN RATF
250
CHANGE ANY REGION OF ZEROS THAT IS TOO SMALL TO ALL ONES IAW TABLE 5
260
IS THE FIRST REGION IN RATF ZEROS?
YES
NO
270
FIND DISTANCE BETWEEN THE CENTERS OF THE FIRST TWO REGIONS OF ZEROS, CALL THIS RDCLEN; IF THERE IS NO SECOND REGION OF ZEROS SET RDCLEN = 24
280
FIND DISTANCE BETWEEN THE CENTERS OF THE FIRST TWO REGIONS OF ONES, CALL THIS RDCLEN; IF THERE IS NOT A SECOND REGION OF ONES SET RDCLEN = 24
290
IS RDCLEN > 12 AND RTAPDLY NOT MAX?
YES
NO
300
RTAPDLY = RTAPDLY + 1
310
SET RTAPSAM SO THAT WILL SAMPLE AT THE CENTER OF THE LARGEST REGION OF ZEROS
FINISHED ACQUISITION SO START TRACKING (FICURE 11)

Figure 7. Process used during the acquisition phase.

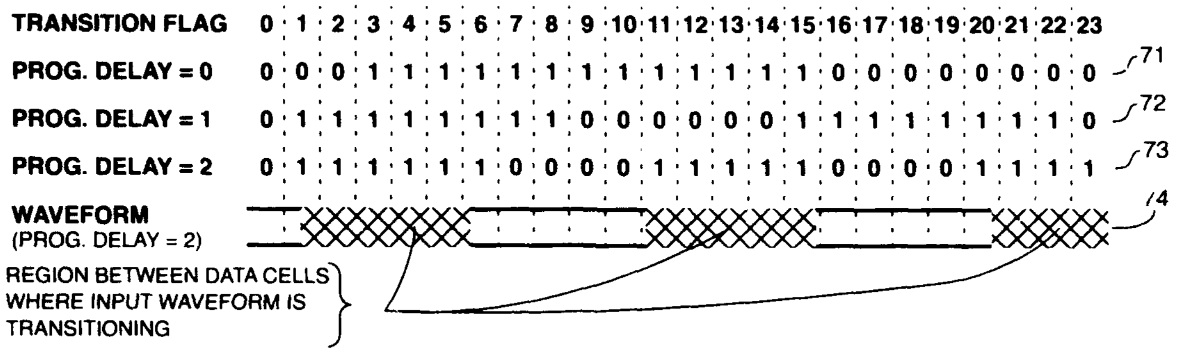

TRANSITION FLAG 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
PROG. DELAY = 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
PROG. DELAY = 1 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0
PROG. DELAY = 2 0 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1
71
72
73
74
WAVEFORM (PROG. DELAY = 2)
REGION BETWEEN DATA CELLS WHERE INPUT WAVEFORM IS TRANSITIONING

Figure 8. Change in transition flags as programmable delays are changed.

TABLE 5
Deletion of Small Regions of Zeros for the Illustrated Embodiment

Size of largest region of zeros	Delete regions of zeros this size or smaller
1	—
2	—
3	1
4	1
5	2
6	2
7	3
8	3
9	4
10	4
11	5
12	5
13	6
14	6
15	7
16	7

2.3 GUARANTEEING ENOUGH FRAMES FOR ACQUISITION AND TRACKING

When idle symbols are not transmitted between frames, it is important to ensure sufficient incoming frames to acquire the signal in a timely manner during initialization and fault recovery. It is also important to have enough frames across the link to be able to track any changes, as will be discussed later. Figure 9 and Figure 10 give flow diagrams for the processes in the illustrated embodiment, which keep track of link status and make sure that the required minimum levels of traffic are present. Figure 9 is the process associated with transmission and Figure 10 is the process associated with the reception of frames. Although these figures show some of the logic involved in sending and receiving data frames (frames that are doing more that just link house keeping), they do not attempt to show all the logic that would be used for flow control and the retransmission of frames in response to transmission errors. They also do not include the logic associated with the transmission and reception of an "update programmable delays frame sequence," which will be discussed later.

When the system is first powered-up or recovering from a fault, steps 610, 620, 630, 640, 650, 660, and 710 (Figure 9) keep transmitting bursts of frames. These are frames of the ACK frame format (Figure 5) with their 2-bit ACKOP field set to zero, indicating an NOP (no-operation). The bursts will continue to be sent until TOK comes on, indicating that the link is up (the link has to be up in both directions for TOK to come on). While the remote end is going through the acquisition process described earlier, it is best to transmit frames continuously. The length of the burst is determined by TAQLEN. The length is chosen such that under normal conditions the link will come up well before the burst has gone on for TAQLEN. Once TOK comes on the burst will be aborted. If the transmission media is inoperative, the entire burst will be transmitted and then the transmitting end will wait for a length of time specified by TAQINTV before transmitting another burst. The transmitting end will keep alternating bursts with pauses until the link comes up. In the illustrated embodiment, bursts with pauses are used instead of continuous transmission to conserve power in the event the link is inoperative due to a hard failure.

The logic associated with retransmission of frames (not shown in figures) will turn off TOK if there are too many consecutive transmission errors. Too many is defined by the register named TCERRCOK (Table 3). If there is no ACK or NAK to a data frame (a frame of the general frame format that is carrying data) within a time specified by the register TTO (transmit time out), this occurrence will be counted as a transmission error. The reception of a NAK will also be considered a transmit error. TCERRCNT (Table 3) is a count of the number of consecutive transmit errors. Also, if there are not frames received for longer than the interval defined by the register TLTO (transmit link time out), TOK will be turned off.

In order for the remote end to track phase changes of the incoming data stream, a minimum level of traffic is required. Steps 680, 690, and 700 guarantee this traffic by transmitting an NOP frame whenever nothing has been transmitted for longer than the interval defined by the register TKAINTV.

Figure 10 shows the process at the receive end of a link. If FCS (frame check sequence) is OK on a received frame, step 840 will zero the count of consecutive errors (RCERRCNT). Steps 850 and 860 turn on the ROK bit and send a frame to the remote end that serves both as an ACK and to update the status of the TOK bit (on the remote end). Steps 870, 880, and 940 update the TOK bit at the local end in response to the state of the ROK bit at the remote end (as communicated via incoming frames). If there are too many consecutive receive errors (either incorrect FCS or time-out), steps 920 and 930 turn off ROK and put the data sampling logic from tracking mode back into acquisition mode.

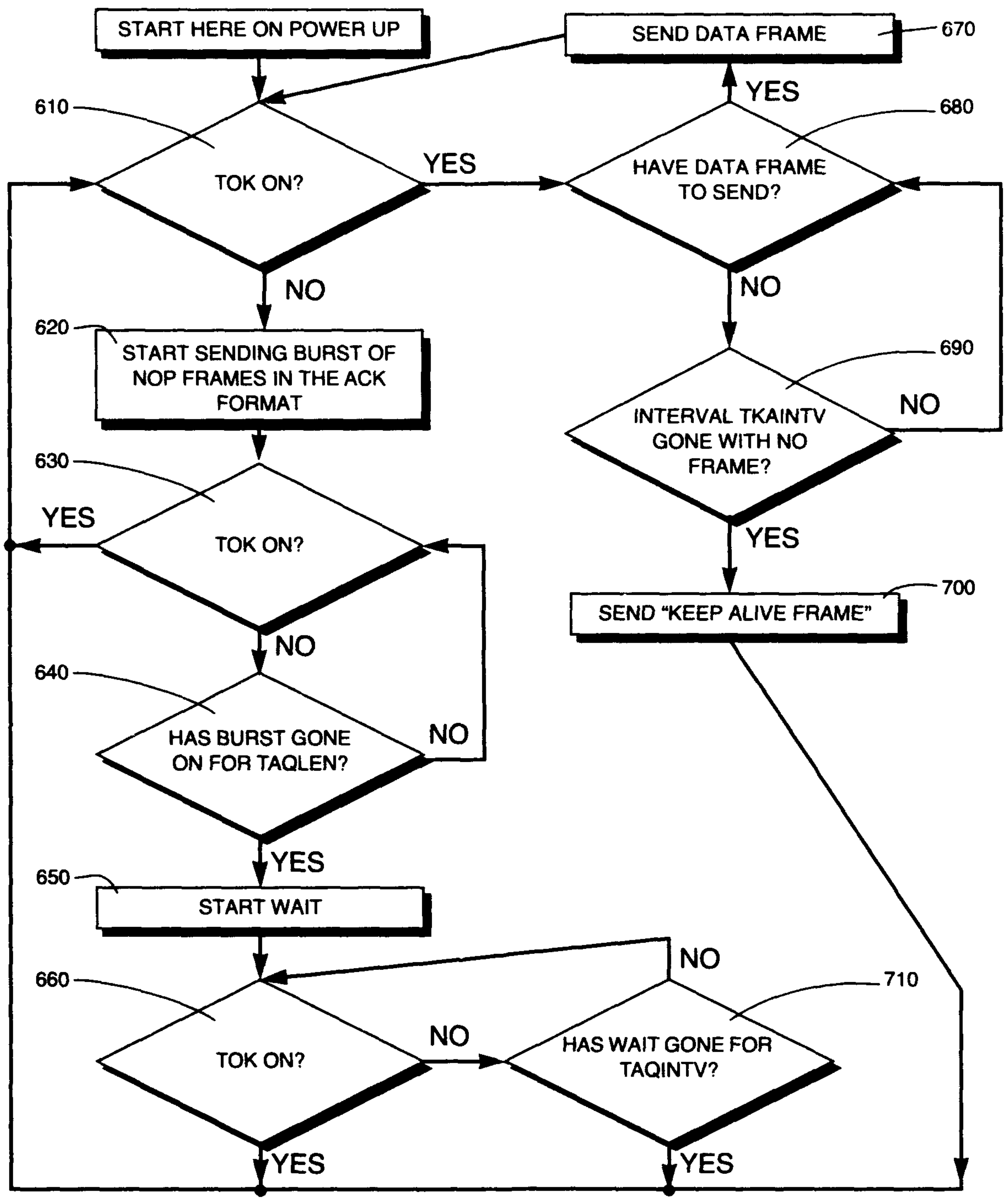


Figure 9. Process on transmit side of link used to keep track of link status.

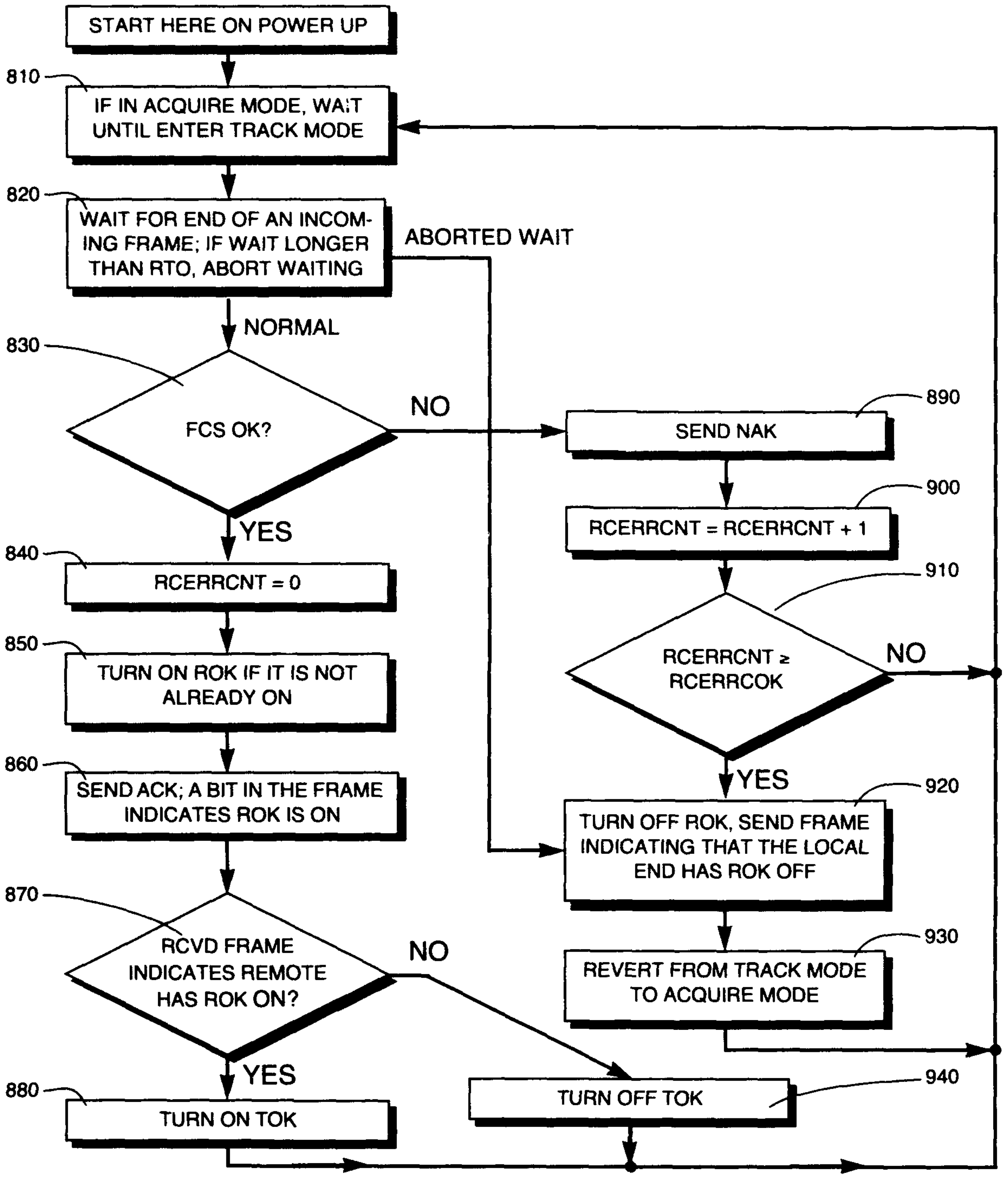

START HERE ON POWER UP
810
IF IN ACQUIRE MODE, WAIT UNTIL ENTER TRACK MODE
820
WAIT FOR END OF AN INCOMING FRAME; IF WAIT LONGER THAN RTO, ABORT WAITING
ABORTED WAIT
NORMAL
830
FCS OK?
NO
YES
840
RCERRCNT = 0
850
TURN ON ROK IF IT IS NOT ALREADY ON
860
SEND ACK; A BIT IN THE FRAME INDICATES ROK IS ON
870
RCVD FRAME INDICATES REMOTE HAS ROK ON?
NO
YES
880
TURN ON TOK
890
SEND NAK
900
RCERRCNT = RCERRCNT + 1
910
RCERRCNT ≥ RCERRCOK
NO
YES
920
TURN OFF ROK, SEND FRAME INDICATING THAT THE LOCAL END HAS ROK OFF
930
REVERT FROM TRACK MODE TO ACQUIRE MODE
940
TURN OFF TOK

Figure 10. Process on receive side of link used to keep track of link status.

2.4 TRACKING THE INCOMING SIGNAL

Figure 11 shows the process executed by the state machine 47 during its tracking mode, which is used after the acquisition mode has completed the initialization or fault-recovery process. Figure 12 is the process followed to decide whether to update the amount of delay provided by each tap of the programmable delay line; this decision is step 430. Figure 13 is the process for implementing step 520. In other words, Figure 12 and Figure 13 are subprocesses to the main process in Figure 11.

During the acquisition mode the state machine is not aware of the frame boundaries, so all incoming symbols, including idle symbols, are accumulated for finding the transition regions, with the exception of the first bytes after a period of nothing being transmitted. During tracking mode, accumulation is only done over data cells from frames that are verified correct by checking the corresponding FCS (frame check sequence). This way the tracking process will not try to track noise during momentary outages. If it is not practical to check the FCS in a particular embodiment, the state machine could accumulate over all symbols except for the first ones after a period of no data. When the FCS is not being checked, accumulation can also take place over idle symbols.

Step 410 (Figure 11) initializes registers RACNT (count of number of bytes have accumulated over) and RATF (accumulate transition flags) in preparation for accumulation of transition flags over multiple frames. Step 420 clears the transition flags, waits for the beginning of a frame, and starts the accumulation of a single frame. Step 430 decides whether to update the value in the register RTAPDLY, which controls the propagation delay provided by each of the programmable delays. Step 430 also waits for the end of the frame. The process for making this decision is given in Figure 12. Step 431 of Figure 12 checks to see if either of the two flags indicating that programmable delays should be changed (either up or down in propagation delay) are set. If neither flag (RINCPD or RDECPD) is set, the decision is made not to change the programmable delays. The flag RWUPD indicates that if the programmable delays are going to be changed, an "update programmable delays frame sequence" should be used. Step 433 checks the header of the frame currently coming in to see if it is the first frame of an update programmable delays frame sequence. If it is, step 436 will wait for the end of this frame and step 437 will make sure the FCS is OK.

Step 450 (Figure 11) checks the FCS. If the FCS is OK, step 460 will OR the transition flags resulting from the previous frame into RATF (accumulate transition flags). Step 470 checks to see if enough bytes have been accumulated over.

If step 430 decides to change the register RTAPDLY, step 490 updates RTAPDLY, changing the propagation delay provided by the programmable delays. Step 490 also initializes RACNT, RATF, and the transition flags in preparation for accumulation. Step 500 accumulates over RADCNTCD byte times. Step 510 ORs the result of step 500 into the register RATF. Step 520 executes the process given in Figure 13 in order to update RTAPSAM (which tap from the delay line is being selected by multiplexer 23).

In Figure 13, steps 521, 522, 523, 524, and 525 find the length of a data cell in terms of the number of taps in the same way that steps 240, 250, 260, 270, and 280 (Figure 7) do during the acquisition process. Step 526 uses the criteria given in Table 6 to decide if it should set a flag (RINCPD or RDECPD) indicating a desire for changing RTAPDLY. If RINCPD or RDECPD are being set and the flag RWUPD is set, step 527 will send a frame requesting the remote end to send an update programmable delays frame sequence. Steps 528 and 529 wait until the end of the current frame (if there is one) then update the RTAPSAM, which causes the multiplexer to select the tap in the center of the largest region of zeros.

In the illustrated embodiment, the programmable delays are changed just after the end of a frame if they are going to be changed at all. After the programmable delays have been changed, it will frequently be impossible to receive good data until after the multiplexer selection has been updated to select the appropriate tap in the programmable delay line. To prevent the loss of real data, the illustrated embodiment will be operated with the bit RWUPD true. With this bit true, the receiver will send a frame to the transmitting end requesting it to send an update programmable delays frame sequence whenever it detects the need for the programmable delays to be changed. The transmitting end will respond with an update programmable delays frame sequence, which consists of: an ACK frame with the ACKOP field set to UPD, indicating that this is the first frame of an update programmable delays frame sequence; and enough ACK frames, with ACKOP set to NOP, to allow the receiving end to accumulate transitions over the number of bytes specified by RADCNTCD (Table 4). The number ACK frames—with ACKOP set to NOP—transmitted as part of an update programmable delays frame sequence is specified by the register TUPCNT (Table 3).

The act of changing the multiplexer 23 (Figure 1) might put a glitch on the data stream. Changing the multiplexer could also cause a data cell or cells to be repeated or skipped by jumping over them in the tapped delay line. Step 528 waits until the end of a frame before changing the tap selected by the multiplexer. Frames start with at least a two-byte preamble; even if a frame immediately follows the one after which the multiplexer selection is changed, at worst some of the preamble will not be received properly.

In operation, the setting of the programmable delays is not expected to change very often after the system has reached thermal equilibrium. When the programmable delays are changed, either transmission time is given up while an update programmable delays frame sequence is transmitted or data will probably be lost while the state machine decides where to sample the incoming data stream. For this reason the criteria for changing the programmable delays is preferably very conservative. It tries to maintain the value of RDCLEN between 8 and 12 taps of the delay line but gives oscillation avoidance a higher priority than staying between 8 and 12.

For very low settings of the programmable delays, the jump-in delay (Table 2) from one setting to the next is very large. If the criteria for changing the programmable delays were chosen to try to keep RDCLEN between 8 and 12, the state machine might oscillate between two settings of the programmable delay lines. For example, assume a signal with a data cell length of 5 ns (i.e., baud rate of 200 Mhz); if with RDCLEN equal to 7 the programmable delays were changed from a setting of 1 to 0, RDCLEN would increase to 1.8 (ratio of delay for $N = 1$ to $N = 0$) $\times 7 = 12.6$. The state machine might measure RDCLEN to be greater than 12 (which it is) and then decide to increment the setting of the programmable delay lines. Hence the programmable delay lines would keep oscillating between a setting of 0 and 1. This problem is solved by having different criteria for changing the setting programmable delays for low settings of the programmable delays (see Table 6). Another solution would be to have finer control on the lowest settings of the programmable delays; this was not practical with the circuit elements available and the requirement to operate at the highest practical baud rates.

Figure 14 gives some examples of how a waveform with RDCLEN = 12 might look in the tapped delay line. Waveform 81 has a region of zeros that is only 1 tap wide. This signal is marginal—if the setting of programmable delays were increased, the region of zeros might end up being less than one tap in width, making it impossible to receive reliably. Although the signal with the region of zeros equal to 1 is marginal, it shows the operation of the illustrated embodiment under extreme conditions. In this example the transition region between two data cells is 11 taps wide. Waveforms 82, 83, and 84 illustrate what other waveforms with RDCLEN = 12 might look like.

Figure 15 gives some examples of how a waveform with RDCLEN = 24 might look in the tapped delay line. The state machine will try to get two complete data cells in the delay lines. In cases such as shown in waveform 92 of Figure 15, where there is a single region of zeros 95 near the center, the state machine cannot measure the width of a data cell. In this case, step 524 (Figure 13) will be trying to measure the distance between the centers of two regions of zeros; since it will not find a second region of zeros, it will set RDCLEN equal to 24 (it is just by happenstance that in this example 24 is the correct value). If possible, the state machine will increase the value of the programmable delays. If the programmable delays are already at their maximum setting, in the case of waveform 92, the state machine will still be able to find the center of a region of zeros, properly sampling the data stream.

Waveform 94 in Figure 15 has a "region of zeros" 96 that is 12 taps wide and is split into two subregions; therefore, the state machine will see it as two different regions of zeros, one 7 taps wide and one 5 taps wide. The sampling point will be picked as the output of tap 3. The optimum sampling point, at the center of the data cell, would be the output of either tap 0 or 1. Although the optimum decision is not being made, there should be no trouble receiving good data in this situation. In this situation it is important to accumulate transitions over enough data (i.e., RADCNT large enough) that the actual regions of transitions are not significantly larger than the ones that the state machine finds. If the accumulation is over a small number of bytes, it is possible that the full extent of the region of transitions will not be found; as a consequence, the sampling point picked could be within the region of transitions, causing data to be lost.

Assume that the illustrated embodiment is fabricated with circuit elements that have a variation in their propagation delays of between .65x and 1.8x over process variation, temperature, and supply voltage. The minimum reliable baud rate, under worst case conditions, will have RDCLEN = 24 with the programmable delays at their maximum settings and the propagation delay of the underlying components at their minimum (factor of .65x typical). This comes out to be: 24 (RDCLEN) x 4.09 ns (max programmable delay setting) x .65 (factor for variation in propagation delay of circuit elements) = 63.8 ns, which corresponds to 15.7 MHz. In reality, one could operate more slowly so long as the region of zeros is large enough that some of it is always in the delay line. Operation with RDCLEN greater than 24 should be avoided.

If the region of zeros is more that 25% of the data cell, a waveform with RDCLEN = 4 could be properly sampled. The maximum baud rate, under worst case conditions, for which this is true is: 4 (RDCLEN) x .4 ns (minimum programmable delay setting) x 1.8 (factor for variation in propagation delay of circuit elements) = 2.88 ns, which corresponds to 347 MHz. This may not be the limiting factor. The critical path is the propagation time out of the D flip-flops 27 (Figure 1) through the XOR gates 41

(Figure 3), followed by the setup time to the JK flip-flops 43 (Figure 3). This comes to (1.38 ns + .51 ns + .44 ns) × 1.8 = 4.19 ns, which corresponds to 238 MHz.

Note that the numbers used for propagation delays in the embodiment given here are not based on an actual layout of the circuit elements presented. When the circuit is actually laid out, the capacitive loading will change the numbers somewhat. The numbers given here are intended to be representative for the illustration of principles, though. It is intended that the illustrated embodiment, if fabricated in a process corresponding to the numbers given here, could be operated reliably with data rates of between 20 and 200 Mbaud.

TABLE 6
Criteria for Changing Programmable Delays for the Illustrated Embodiment

Value of RTAPSAM	Value of RDCLEN	Action†
RTAPDLY = 1	RDCLEN < 5	Set DECPD
RTAPDLY = 2	RDCLEN < 6	Set DECPD
RTAPDLY = 3	RDCLEN < 7	Set DECPD
RTAPDLY = 4 OR RTAPDLY > 4	RDCLEN < 8	Set DECPD
RTAPDLY < 9 OR RTAPDLY = 9	RDCLEN > 12	Set INCPD

* This table is referenced as part of the process specified by Figure 13.

† The action is taken if both the conditions for the value of the RTAPSAM and the value of RDCLEN are met.

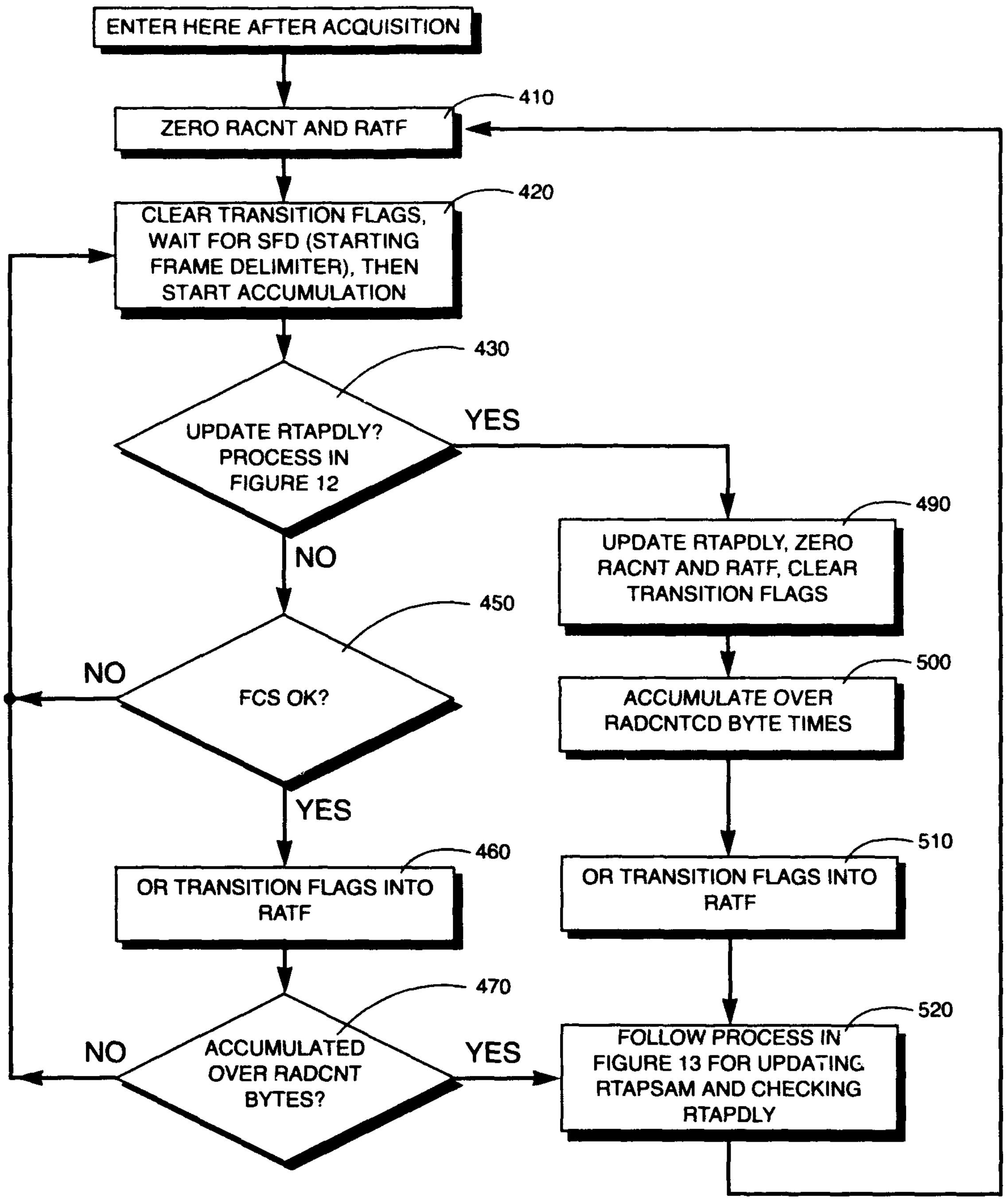
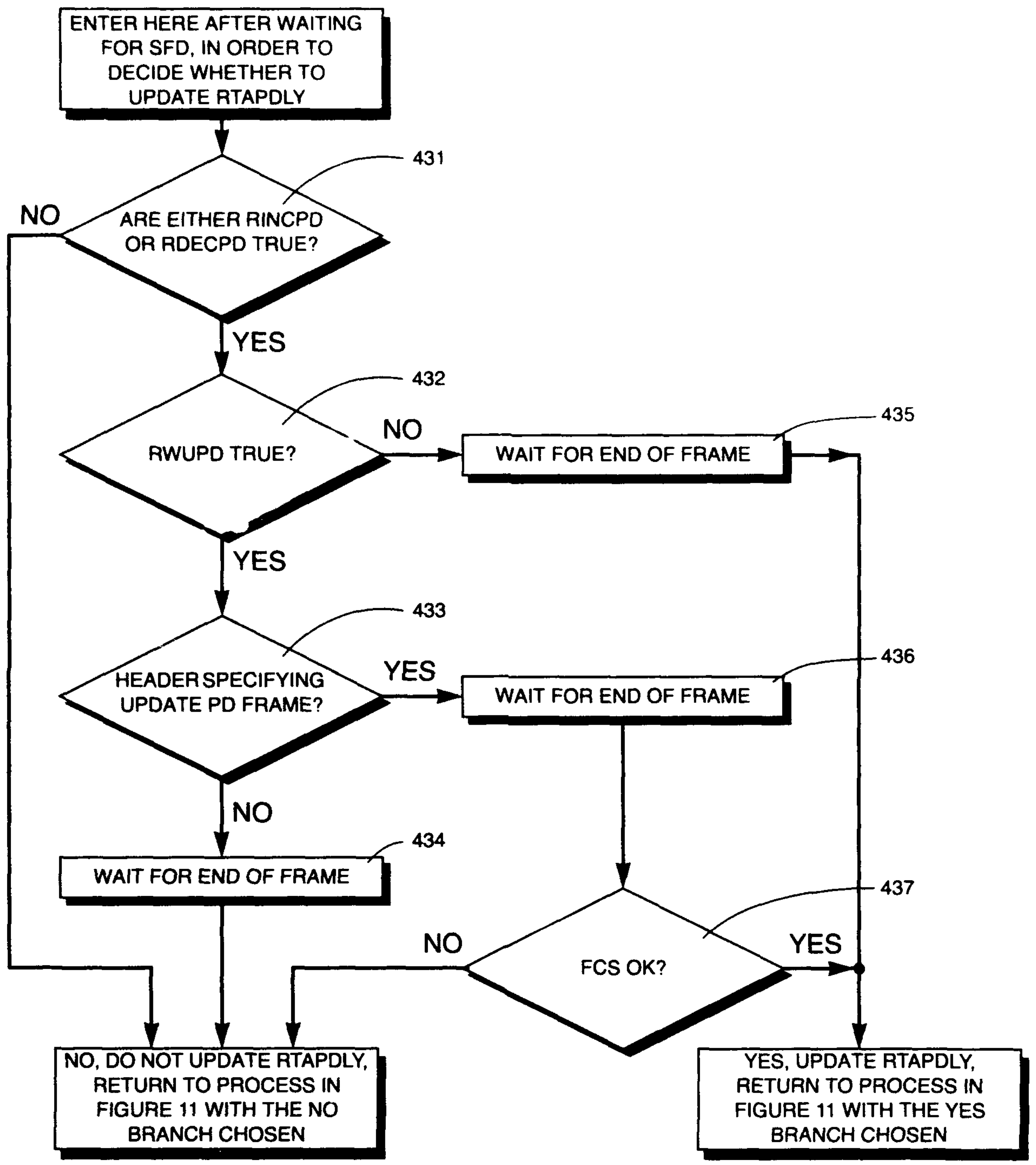

ENTER HERE AFTER ACQUISITION
410
ZERO RACNT AND RATF
420
CLEAR TRANSITION FLAGS, WAIT FOR SFD (STARTING FRAME DELIMITER), THEN START ACCUMULATION
430
UPDATE RTAPDLY? PROCESS IN FIGURE 12
YES
NO
490
UPDATE RTAPDLY, ZERO RACNT AND RATF, CLEAR TRANSITION FLAGS
450
NO
FCS OK?
YES
500
ACCUMULATE OVER RADCNTCD BYTE TIMES
460
OR TRANSITION FLAGS INTO RATF
510
OR TRANSITION FLAGS INTO RATF
470
NO
ACCUMULATED OVER RADCNT BYTES?
YES
520
FOLLOW PROCESS IN FIGURE 13 FOR UPDATING RTAPSAM AND CHECKING RTAPDLY

Figure 11. Process used for tracking phase.



ENTER HERE AFTER WAITING FOR SFD, IN ORDER TO DECIDE WHETHER TO UPDATE RTAPDLY
431
ARE EITHER RINCPD OR RDECPD TRUE?
NO
YES
432
RWUPD TRUE?
NO
435
WAIT FOR END OF FRAME
YES
433
HEADER SPECIFYING UPDATE PD FRAME?
YES
436
WAIT FOR END OF FRAME
NO
434
WAIT FOR END OF FRAME
437
FCS OK?
NO
YES
NO, DO NOT UPDATE RTAPDLY, RETURN TO PROCESS IN FIGURE 11 WITH THE NO BRANCH CHOSEN
YES, UPDATE RTAPDLY, RETURN TO PROCESS IN FIGURE 11 WITH THE YES BRANCH CHOSEN

Figure 12. Process used to update RTAPDLY, called by 430 in Figure 11.

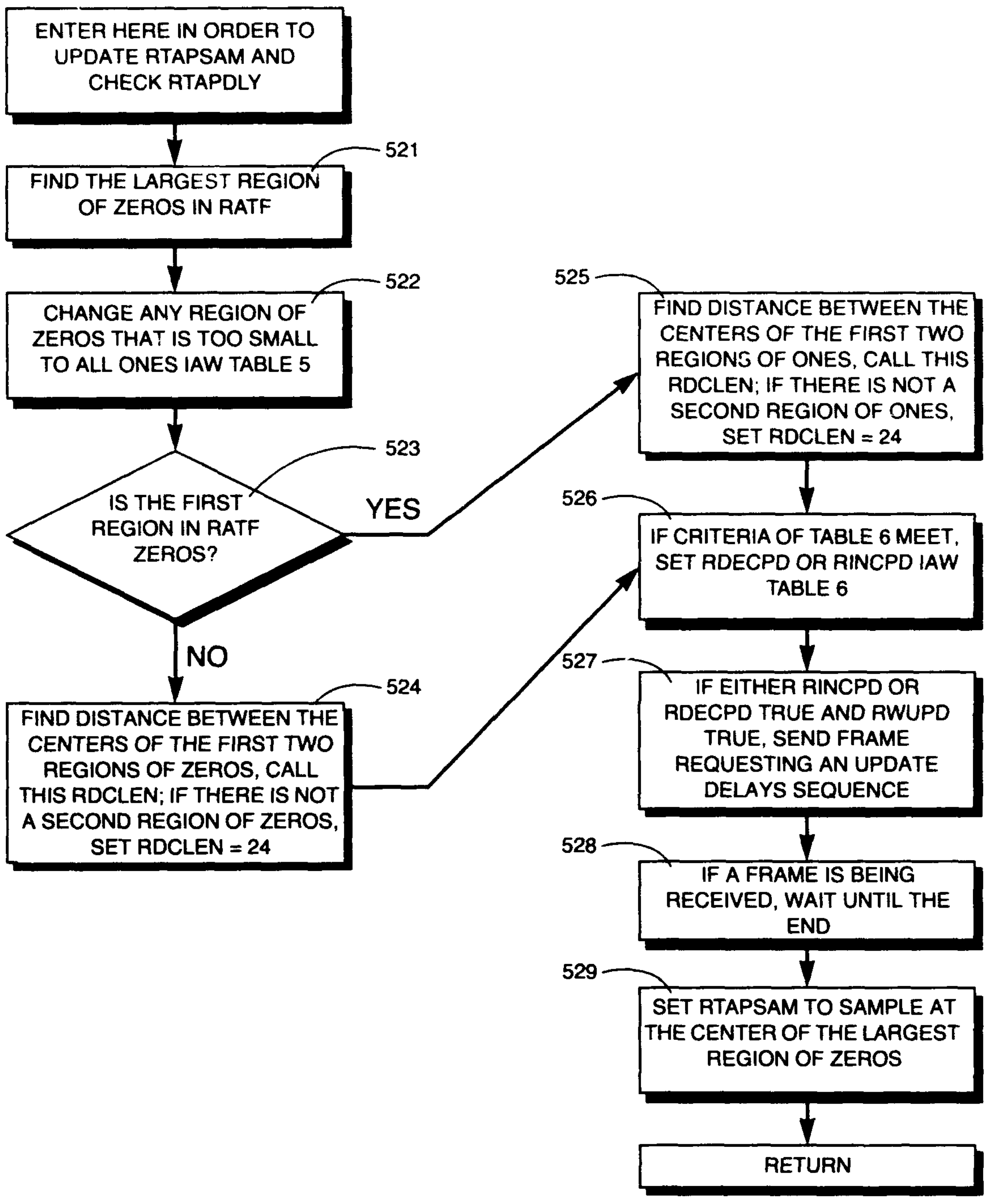

ENTER HERE IN ORDER TO UPDATE RTAPSAM AND CHECK RTAPDLY
521
FIND THE LARGEST REGION OF ZEROS IN RATF
522
CHANGE ANY REGION OF ZEROS THAT IS TOO SMALL TO ALL ONES IAW TABLE 5
523
IS THE FIRST REGION IN RATF ZEROS?
YES
NO
524
FIND DISTANCE BETWEEN THE CENTERS OF THE FIRST TWO REGIONS OF ZEROS, CALL THIS RDCLEN; IF THERE IS NOT A SECOND REGION OF ZEROS, SET RDCLEN = 24
525
FIND DISTANCE BETWEEN THE CENTERS OF THE FIRST TWO REGIONS OF ONES, CALL THIS RDCLEN; IF THERE IS NOT A SECOND REGION OF ONES, SET RDCLEN = 24
526
IF CRITERIA OF TABLE 6 MEET, SET RDECPD OR RINCPD IAW TABLE 6
527
IF EITHER RINCPD OR RDECPD TRUE AND RWUPD TRUE, SEND FRAME REQUESTING AN UPDATE DELAYS SEQUENCE
528
IF A FRAME IS BEING RECEIVED, WAIT UNTIL THE END
529
SET RTAPSAM TO SAMPLE AT THE CENTER OF THE LARGEST REGION OF ZEROS
RETURN

Figure 13. Process used to update RTAPSAM, called by 520 in Figure 11.

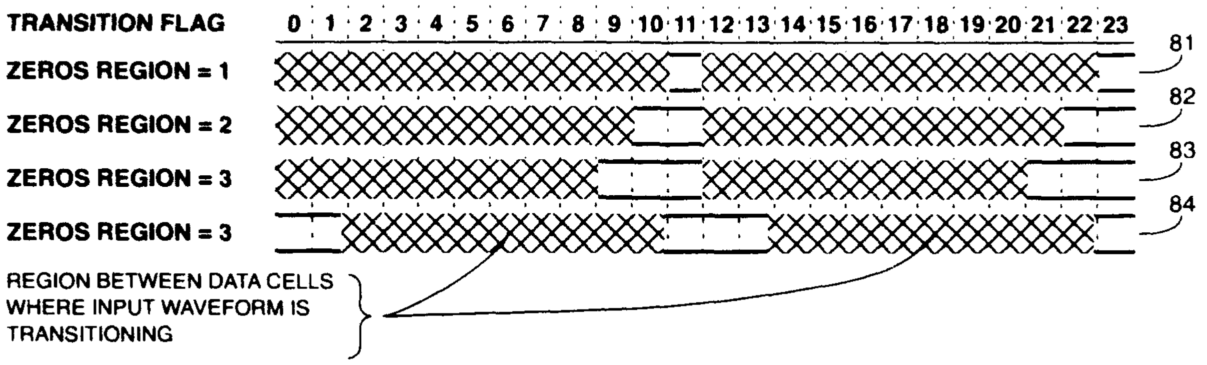


Figure 14. Transition flags that might be set in response to data cells equal to 12 taps.

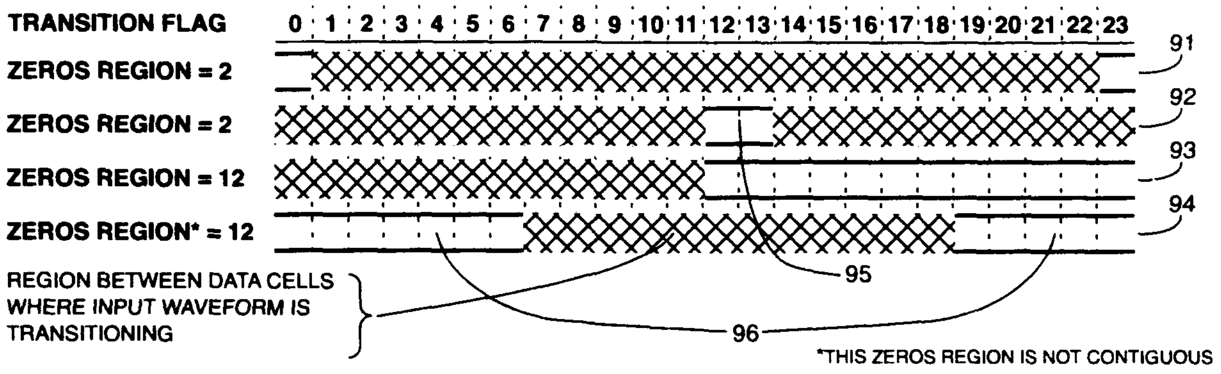


Figure 15. Transition flags that might be set in response to data cells equal to 24 taps.

3. ADDITIONAL EMBODIMENTS

Some systems may have a totally different phase for each frame but a data clock that is nearly the same frequency. A system with a single data clock and multiple potential transmitters on the same bus is an example. In this case the state machine could be designed to accumulate transitions over some of the preamble and use this information to set the sampling point before the start of the actua data. Accumulation to update the value of RTAPDLY (the propagation setting for the programmable delays 25_0 through 25_{N-1}) could be done during the data portion of the message with the programmable delays being updated just after the end of a frame—so that changing the programmable delays does not cause any data to be lost.

In a system with multiple potential transmitters where all the transmitters are at the same data rate, a multiplexer can be used to switch the input from the received data stream (10 Figure 1) to a calibration waveform. The calibration waveform is then used to set the value of RTAPDLY. The process of setting RTAPDLY with a calibration waveform instead of data is much simpler—the calibration waveform should have almost no jitter. This technique, as well as the one for tracking phase described in the next paragraph, can be used to receive Ethernet data. For information on obtaining further documentation concerning an embodiment for Ethernet, see Appendix A.

The illustrated embodiment is designed under the assumption that phase changes between the data and its clock are relatively slow—the phase does not change significantly over multiple frames. In a system wherein the phase is changing more rapidly, the state machine could be designed to accumulate over a fraction of a frame and then update the sampling multiplexer immediately instead of waiting until the end of a frame. With such a scheme, the sampling point might be updated many times during the course of a single frame. In this case it would be important to change the sampling point in such a way that the data stream is not disrupted. This is not a problem with the illustrated embodiment because that embodiment waits until just after the end of a frame to change the sampling point.

In a scheme where the sampling point is updated many times during a single frame, tracking of the delay being provided by each tap of the delay line can be accomplished by updating the value of RTAPDLY just after the end of a frame, using the preamble of the next frame if necessary to adjust the sample point. Alternatively, sample point adjustment could be performed during the frame after a change kTAPDLY. This would require accepting a loss of data, knowing that the protocol will request retransmission of lost frames. RTAPDLY should not need to be changed very often.

In the case of a system in which the data rates are significantly different for each frame, the state machine could be designed to reacquire the signal for each frame. In this case, the value of RTAPDLY and the sampling point could both be selected during the preamble.

In a system that is not checking the FCS before using accumulation data, it may be desirable to make the changing of the programmable delay lines more immune to noise. This can be done by requiring the state machine to make N consecutive decisions in a row (all pointing to the programmable delay lines being changed in the same direction) before the programmable delay lines are actually changed.

In systems where it is desirable to save power, a data detect circuit can be added to detect whether there is data coming in on signal 10 (Figure 1). This data detect circuit could be used to gate the data clock 11 off to the remainder of the circuit when no data is coming in. The data detect circuit may be implemented using a copy of this clock that is not gated off.

APPENDIX A
LICENSING INFORMATION

There is a patent pending for this device. Assuming that the patent is granted and the title (of the patent) does not change between now and when it is granted, the title will be "Data Sampling Circuit for a Burst Mode Communication System." Those interested in obtaining a license to MIT's patent rights should reference MIT Case No. 6122L and contact:

MIT Technical Licensing Office
28 Carleton St., Room E32-300
Cambridge, MA 02139
MIT case #6122L
617-253-6966

A document is available that gives a detailed description of an embodiment of this device adapted to Ethernet. To obtain a copy, contact the MIT Technical Licensing Office at the above address.

There may be some consulting services available for adapting this device to your particular needs. For additional information, contact the MIT Technical Licensing Office at the above address.

REFERENCES

1. A.G. Rocco, P.D. Linton, and J.K. Noonan, "Overview of Enhanced Data Stream Array Processor (EDSAP)," MIT Lincoln Laboratory, Lexington, Mass., Technical Report 974, 6 May 1993.
2. A.S.G. Ramakrishnan, "FDDI: A High Speed Network", PTR Prentice-Hall Inc., 1994.
3. J.S. Butcher, U.S. Patent No. 4 789 996 (6 December 1988).
4. B.J. Nelson, U.S. Patent No. 4 819 251 (4 April 1989).
5. R.H. Leonowich and J.L. Sonntag, U.S. Patent No. 5 040 193 (13 August 1991).

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE 1 September 1994	3. REPORT TYPE AND DATES COVERED Technical Report
4. TITLE AND SUBTITLE Data Sampling for a Burst Mode Communications System		5. FUNDING NUMBERS C — F19628-90-C-0002 PR — 224
6. AUTHOR(S) A. Gregory Rocco		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lincoln Laboratory, MIT P.O. Box 73 Lexington, MA 02173-9108		8. PERFORMING ORGANIZATION REPORT NUMBER TR-1012
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory Griffiss AFB, NY 13441-4514		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-94-096
11. SUPPLEMENTARY NOTES None		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

A data sampling circuit and method is provided for a burst mode communication system. The circuit is an entirely digital circuit for reliably sampling an incoming stream of data to automatically adjust to variations in data stream clock rates and phase variations in the incoming data. The circuit includes a delay line with a plurality of serially coupled taps, each tap having a variable delay. One aspect of the invention includes increasing the delay time until the delay line captures at least one, but preferably two, full data cells of the incoming data stream (i.e., signal levels over at least one full clock period, defined by two transitions of the data stream), thereby aligning the receiving circuit with the frequency of the data stream clock. A second aspect of the invention includes outputting data from a tap that is selected to be midway between two regions of transitions of the incoming data stream, thereby aligning the receiving circuit with the phase of the data stream clock. The invention can, in alternative embodiments, track changes in the input data stream's phase/frequency. This involves updating the amount of delay in each tap of the delay line as well as picking the output of the appropriate tap to be used as the sampled data stream in response to changes in the input signal as well as changes in propagation delays of the circuits used to implement the delay line, resulting from temperature and voltage variations.

14. SUBJECT TERMS			15. NUMBER OF PAGES 52
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Same as Report